# Multi-agent Positioning Mechanism in the Dynamic Environment

Hidehisa Akiyama and Itsuki Noda

Information Technology Research Institute,
National Institute of Advanced Industrial Science and Technology
Ibaraki, Japan
{hidehisa.akiyama,I.Noda}@aist.go.jp

**Abstract.** In this paper, we propose a novel agent positioning mechanism for the dynamic environments. In many problems of the real-world multi-agent/robot domain, a position of each agent is an important factor to affect agents' performance. Because the real-world problem is generally dynamic, a suitable positions for each agent should be determined according to the current status of the environment. We formalize this issue as a map from a focal point like a ball position in a soccer field to a desirable positioning of each player agent, and propose a method to approximate the map using Delaunay Triangulation. This method is simple, fast and accurate, so that it can be implemented for real-time and scalable problems like RoboCup Soccer. The performance of the method is evaluated in RoboCup Soccer Simulation environment compared with other function approximation method like Normalized Gaussian Network. The result of the evaluation tells us that the proposal method is robust to uneven sample distribution so that we can easily to maintain the mapping.

## 1 Introduction

In many problems of the multi-agent/robot domain in real-world environment, the position of each agent is a significant factor to affect the agent's performance. For example, in a multi-robot transportation system, a cordinatin of positionings of robots are decided carefully by given tasks and status inputs in order to realize efficient convey. Similarly, in the games which teams play in the dynamic environment like a soccer, each agent should make a decision about its positioning continuously according to the current game status in order to fulfill its role or duty. Otherwise, the performance of the team will be decreased significantly.

In order to realize effective positionings or formations of agents, most of RoboCup teams generally use rule-based and/or numerical-function-based position definition systems. However, this approach meets a difficulty when we like to tune the positionings for a certain situations like case when a ball is in a penalty area. Because penalty areas are very important region in soccer, small miss of the position may cause critical situation. Therefore, rules and functions to determine the position in such cases becomes so complex that it becomes difficult to maintain.

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 377–384, 2008.

Another approach for the positioning is to utilize machine learning methods. In this approach, we just need to give examples of desireble positioning or training schema for agents to learn suitable positioning rules/functions. For example in the penalty-area cases above, we just give many examples for penalty area to tune sensitive positioning of the dangerous region. However, this learning approach has a general problem how to give enough example or training for agent. Especially in a complex domain like RoboCup, it is difficult to give enough number of examples for learning by hand. It is also difficult to provide reasonable evaluation method for trainings to provide suitable index of the learning.

In this paper, we propose a novel positioning mechanism which uses Delaunay Triangulation and the linear interpolation method. In this method, we formalize learning problem of the positioning as a function approximation. The mechanism used in the model is effective to acquire the interpolation function, which a focal point in the environment is used as input and agents' move target position is output. In the experiment, we used the RoboCup Soccer Simulator[6,1] as an experiment environment. We compared our method with other function approximation method, Normalized Gaussian Network.

## 2 Positioning Problem in the RoboCup Soccer Simulation

In the RoboCup Soccer Simulation domain, Situation Based Strategic Position (SBSP)[8] is well known as the agents' positioning mechanism. SBSP uses a ball position as a focal point and does not consider other agents. But, if we assume that all agents always pay attention to the ball, the cooperative behavior can be done indirectly. Because it is easy to implement SBSP, almost all teams in the RoboCup Soccer Simulation League use SBSP or the similar model.

SBSP uses a simple function that uses the the ball coordinate value as an input value and outputs the player agent's basic move position. And the attraction parameters and the movable region are defined for each agent and are used to calculate the final output value. Fig.1 shows a basic SBSP algorithm.

`BasePos` means the agent's position when the ball is located at the center of field. `AttractionToBall` means the rate that agent follows the ball move.

```
GetSBSPPosition( Num, BallMove )
  Num : agent number
  BallMove: ball move vector from the center of field
1. BasePos := BasePosition( Num );
2. AttractionToBall := BallAttract( Num );
3. SBSPPos := BasePos + BallMove * AttractionToBall
4. Region := MovableRegion( Num );
5. if Region does not contain SBSPPos
6.   adjust SBSPPos into Region
7. return SBSPPos
```

**Fig. 1.** The basic algorithm of SBSP

`SBSPPos` means the agent's move position. If `SBSPPos` is not contained by `Region`, the `SBSPPos` is adjusted into `Region`.

The problem of SBSP is that the output value depends on the used function. In the basic SBSP algorithm, the characteristic of agent's move also becomes simple because the simple linear function is used. If we want the more complicated agent's move, we need to prepare the several parameter set for each situation. But, it is difficult for us to manage such many parameters and the relation of the situations correctly.

In our previous research[2], we used a non-linear function instead of the SBSP algorithm. To acquire the non-linear function, we used a traditional three layer perceptron as a function approximator and the training data set is created by human supervisor using a GUI tool. As a result, we could acquire the good approximation function that can be used in the real game. However, it is difficult to acquire the completely desired result because the overfitting is a critical problem. So, we concluded that we need a locally adjustable function approximation model.

## 3 Positioning Mechanism Using Delaunay Triangulation

We propose a novel positioning mechanism that uses Delaunay Triangulation and the linear interpolation algorithm. In this model, the region is divided into several triangles based on the given training data, and each training data affects only the divided region where it belongs to. So, the proposal model is locally adjustable.

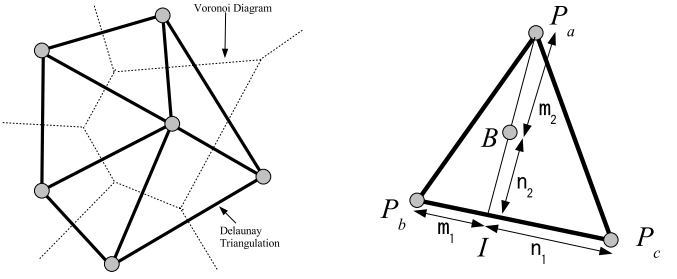
### 3.1 Delaunay Triangulation

Delaunay Triangulation is one of the method to triangulate tha plane region based on the given point set. Delaunay Triangulation for a set $P$ of points in the plane is a triangulation $DT(P)$ such that no point in $P$ is inside the circumcircle of any triangle in $DT(P)$. If the number of given points are more than 3, we can get a unique Delaunay Triangulation. Fig. 2(a) shows the example of Delaunay Triangulation. In this figure, Voronoi Diagram is also shown. There is a duality between Voronoi Diagram and Delaunay Triangulation. In our programs, we implemented the incremental algorithm[4] that is one of the fastest algorithm to calculate Delaunay Triangulation and the time complexity is $O(n \log n)$.

In our proposal method, the ball positions in training data are used as the vertices of triangles and each vertex means the given training data. Each vertex has the output value as an agent's move position for that vertex(=ball) position. When the ball is contained by one triangle, agent's move position is calculated by interpolation algorithm described in next section.

### 3.2 Linear Interpolation Algorithm

We use the simple linear interpolation algorithm to calculate the agent's move position. This algorithm is same as Gouraud shading algorithm[3]. Gouraud




(a) Example of Delaunay Triangulation

(b) Liner interpolation by Gouraud shading algorithm

**Fig. 2.** Delaunay Triangulation and Linear Interpolation

shading algorithm is a method used in the computer graphics domain to simulate the differing effects of light and color across the surface of an object.

Fig. 2(b) shows the process of Gouraud shading algorithm. The output values from vertices $P_a$, $P_b$ and $P_c$ are $O(P_a)$, $O(P_b)$ and $O(P_c)$ respectively. Now, we want to calculate $O(B)$, the output value of the point $B$ contained by the triangle $P_aP_bP_c$. The algorithm is as follows:

1. Calculates $I$, the intersection point of the segment $P_bP_c$ and the line $P_aB$.
2. The output value at $I$, $O(I)$, is calculated as:

$$O(I) = O(P_b) + (O(P_c) - O(P_b))\frac{m_1}{m_1 + n_1}$$

   where $|\overrightarrow{P_bI}| = m_1$ and $|\overrightarrow{P_cI}| = n_1$.
3. $O(B)$ is calculated as:

$$O(B) = O(P_a) + (O(I) - O(P_a))\frac{m_2}{m_2 + n_2}$$

   where $|\overrightarrow{P_aB}| = m_2$ and $|\overrightarrow{BI}| = n_2$.

### 3.3 Advantages

The proposal positioning mechanism has following advantages:

- High approximation accuracy. Our method realizes higher accuracy than other function approximator.
- Locally adjustable. Even if new training data is added or existing data is modified, the triangle region where that data is not contained is never affected.
- Simple and fast running. It is possible to use it in real time.
- High scalability. Even if the considered region is extended or shrinked, it is easy to correspond to the new region without any changes.

– Complete reproducibility. If the training data is same, we can acquire the completely same result.

Especially, the complete reproducibility is an important advantage. Because there is no limitation of the input order of the training data, any training data can be added at any time. This means that human can intervene at any time.

## 4 Experiment

In order to evaluate our method, we compare our method with other function approximator. We developed the simulated soccer team which can use our method and Normalized Gaussian Netrowk(NGnet)[5].

### 4.1 NGnet

This section describes the brief definition of NGnet. NGnet is one of the extended methods of Radial Basis Function(RBF) Network[7]. The network structure of NGnet and RBFNetwork is almost same as the three layer perceptron. But, the Gaussian basis function is used as an activation function instead of the sigmoid function. These methods are known as a locally adjustable function approximator.

The difference between NGnet and RBFNetwork is whether the output of each unit is normalized by the sum of units' output or not. The normalization guarantees the activation of at least one unit for any input value. On the other hand, in the RBFNetwork, if the distance between units is big, output values become almost 0. In the agent positioning problem, it is not preferable that the output values become 0. Because NGnet can solve this, we adopt NGnet as the compared method in this paper.

The output of the unit $i$ in output layer is

$$f_i(x, w) = \frac{\sum_{j=1}^{N} w_{ij}\phi_j(x)}{\sum_{j=1}^{N} \phi_j(x)} \tag{1}$$

where $x$ is an input vector, $w_{ij}$ is the connection weight for unit $i$ from hidden layer to output layer. $\phi_j(x)$ is the output of basis unit $j$ in hidden layer:

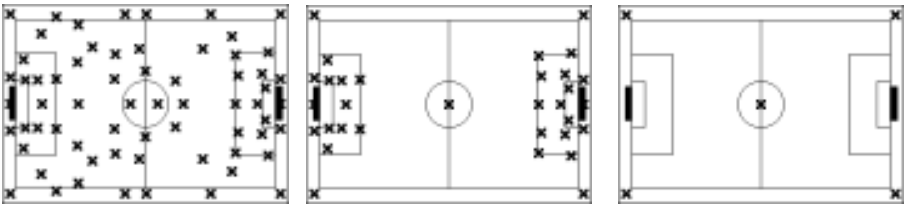
$$\phi_j(x) = \exp(-\frac{||x - c_j||^2}{2\sigma_j^2}) \tag{2}$$

where $c_j$ is the center position of basis unit $j$ and $\sigma_j$ is the variance parameter of basis unit $j$. In NGnet, we need to acquire three parameters, $w_i$, $c_j$ and $\sigma_j$.

The connection weight $w_i$ is learned by the following gradient descent method:

$$w(t+1) = w(t) - \eta\frac{\partial\epsilon}{\partial w} + \alpha(w(t) - w(t-1)) \tag{3}$$

The error value of agent's position at each training data point is used as $\epsilon$ . In this experiment, the learning rate $\eta$ is set to 0.1 and the rate of moment method



(a) All Data (b) Biased Data (c) Minimum Data

**Fig. 3.** Training data set used in the experiment

$\alpha$ is set to 0.5. The ball positions of training data are just used as the center of basis unit $c_j$. So, the number of training data is same as the number of basis unit. The following heuristic value is used as the variance parameter $\sigma$:

$$\sigma = \frac{1}{N} \sum_{i=1}^{N} ||c_i - c_j|| \tag{4}$$

where $c_j$ is the nearest basis unit to the unit $c_i$. All basis units use a same variance parameter.

### 4.2 Experiment Settings

We prepare three training data set which are shown in Fig.3. In each sub figure, 'X' marked points mean the given training data. We used them as the training data for NGnet and the proposal method.

Fig.3(a) shows the normal training data set. In this data set, the training data are given for the whole region. This data set is used by our team TokyoTechSFC in the RoboCup2006 Soccer Simulation League and our team won the 4th place. Fig.3(b) shows the biased training data set. In this biased training data set, several training data are removed from Fig.3(a) and we set the big difference of the density between the field middle area and the penalty area. Fig.3(b) shows the minimum training data set. Almost all training data are removed from Fig.3(b). This is the minimum training data set for the simulated soccer domain.

We developed a simulated soccer team that uses both methods and played games against the fixed team. As an opponent team, we use UvA Trilearn 2004[1] which participated RoboCup2004 and was ranked 7th. For each training data set, one half match[2] is played 50 times respectively.

### 4.3 Results

Table 1 and 2 show the statistics data extracted from log files. Table 1 shows big difference in the average conceded goal between NGnet and the proposal

[1] http://www.science.uva.nl/~jellekok/robocup/

[2] One half is about 3000 cycles.

**Table 1.** This table shows the average scored goal and the average conceded goal. The values in the parenthesis mean the standard deviation for each average value.

| | Goal Scored | | Goal Conceded | |
|---|---|---|---|---|
| | NGnet | Proposal method | NGnet | Proposal method |
| All | 0.2(0.4) | 0.08(0.27) | 0.98(0.89) | 0.76(0.9) |
| Biased | 0.24(0.52) | 0.06(0.22) | 1.76(1.24) | 0.6(0.74) |
| Minimum | 0.02(0.14) | 0.02(0.14) | 3.7(1.97) | 3.16(1.67) |

**Table 2.** This table shows the average number of successful passes, and the average number of successful intercepts. The values in the parenthesis mean the standard deviation for each average value.

| | Successful Pass | | Successful Intercept | |
|---|---|---|---|---|
| | NGnet | Proposal method | NGnet | Proposal method |
| All | 99.06(19.08) | 110.6(18.79) | 20.2(5.06) | 17.18(6.06) |
| Biased | 67.68(15.07) | 91.44(21.35) | 18.8(5.52) | 19.32(5.29) |
| Minimum | 83.86(19.1) | 84.3(18.48) | 12.06(4.17) | 14.5(5.21) |

method with the biased data set. And Table 2 shows that the average number of successful pass becomes minimum in the case of NGnet with the biased data set.

We guess this is because the positions acquired by NGnet with the biased data set has the unique characteristic. When agents uses NGnet with the biased data set, they do not move so much in the field middle area. Even if a teammate agent has the ball, other agents do not move according to the ball move. So, the ball owner agents might not be able to find tha pass courses. On the other hand, in the case of NGnet with all data and minimum data set or the proposal method with any data set, the acquired positions show the smooth move in the whole field area. These characteristic are caused by the variance parameter of NGnet.

These results shows that the density of the training data significantly affects the positioning characteristic of NGnet. As a result, the performance of the team is also easily affected by that characteristic. This means that it is difficult to tune the variance parameter of each basis function unit because the positioning characteristic of NGnet depends on the variance parameter. Therefore, we can say that it is necessary to prepare the uniformly distributed basis function units for NGnet in order to get the stable team performanc, and the proposal method is robust to uneven sample distribution so that we can easily to maintain the mapping.

## 5 Conclusion and Future Directions

Although the statistics data from the experiment may not show the meaningful quantitative result, some criteria show the important characteristic of each method.

We can say that our proposal method has many advantages obviously. However, the proposal method has the following disadvantages:

- The proposal method needs to store all training data. So, it requires many memories.
- The proposal method requires the high cost to keep the consistency of the training data. If one training data is modified, the near training data may be also required to modify.

We plan to develop the method to adjust the training data automatically in order to reduce the management cost of the training data set. At least, we need the method to help us to find the inconsistent training data. And also, we should consider about the multiple dimensional input and output. Now, our proposal method can handle only two dimensional input and output. If we can handle other agents' positions as an input and the other decision making parameters as an output, they will be very useful. We have to consider about the method to compress the dimension or to overlap the information.

## References

1. The RoboCup Soccer Simulator, `http://sserver.sourceforge.net/`
2. Akiyama, H., Katagami, D., Nitta, K.: Team formation construction using a gui tool in the robocup soccer simulation. In: Proceedings of SCIS & ISIS 2006 (2006)
3. Gouraud, H.: Continuous shading of curved surfaces. In: Wolfe, R. (ed.) Seminal Graphics: Pioneering efforts that shaped the field. ACM Press (1998)
4. van Kreveld, M., de Berg, M., Schwarzkopf, O., Overmars, M.: Computational Geometry: Algorithms and Applications, 2nd edn. Springer (2000)
5. Moody, J., Darken, C.: Fast learning in networks of locally-tuned processing units, vol. 1, pp. 289–303 (1989)
6. Noda, I., Matsubara, H.: Soccer server and researches on multi-agent systems. In: Kitano, H. (ed.) Proceedings of IROS 1996 Workshop on RoboCup, pp. 1–7 (November 1996)
7. Poggio, T., Girosi, F.: Networks for approximation and learning, vol. 78, pp. 1481–1497 (1990)
8. Reis, L.P., Lau, N., Olivéira, E.: Situation Based Strategic Positioning for Coordinating a Simulated RoboSoccer Team. Balancing Reactivity and Social Deliberation in MAS, 175–197 (2001)